Fast Dynamic Image Based Lighting for Mobile Realistic AR

Hadrien Croubois*, Jean-Philippe Farrugia†, Jean-Claude Iehl‡
LIRIS - Lyon1 University

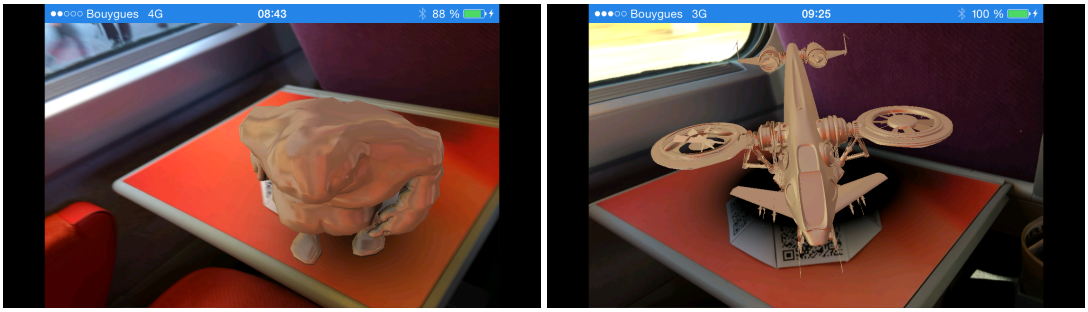

Figure 1: *Real-time mobile realistic augmented reality running real time on a high end mobile phone*

Abstract

This paper presents a mobile implementation of realistic augmented reality using a simple image based lighting method. The front camera of the mobile device is used to interactively capture and update an environment map. Then, by making some reasonable assumptions on local geometry and object reflectance function, incident lighting is integrated in real-time. The method handles dynamic environment and soft shadows, and runs at real-time framerates on high-end devices.

CR Categories: I.3.3 [Computer Graphics]: Three-Dimensional Graphics and Realism—Display Algorithms I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Radiosity;

Keywords: augmented reality, real-time rendering, image-based lighting, environment acquisition

1 Introduction

The term ”realistic augmented reality” (realistic AR) is used to qualify an application where convincing interactions between real and virtual elements are computed. This aspect is important for user immersion: illumination computations add important visual cues to the scene (shadows, object shading, caustics...), notably enhancing global scene comprehension. The main scientific issue of realistic AR is common illumination, which is a common rendering pipeline for virtual and real (acquired) data. Achieving common illumination in a general context is hard: geometric and photometric information of the environment has to be acquired and transformed, potentially at interactive rates, to fit the rendering pipeline. This usually needs specialized hardware (for acquisition) and complex algorithms (for transformation), and is rarely compatible with realtime constraints. Alas, mobile platforms like smartphones or tablets, which are an ideal application playground for realistic AR, have limited computational ressources.

This paper presents an environment map acquisition method and an image-based rendering model that are suitable for mobile realistic AR. The front camera of a mobile device is used to interactively capture and update the environment map. Then, by making some reasonable assumptions on local geometry and virtual object material model, the rendering equation may be simplified enough to analytically integrate incident lighting in real-time by exploiting the filtering capacities of the graphics hardware. The method handles dynamic environment and soft shadows, and is simple enough to be implemented on mobile operating systems. It runs at real-time frame rates on high-end devices.

This document will be organized as follows: following this introduction is a short overview of related work on image-based lighting and realistic AR. The third section will introduce our contributions on environment capture and image-based lighting. The fourth section will present some results, along with limitations and potential extensions in the fifth section. A brief conclusion will end the paper in the sixth section.

2 Related work

Realistic augmented reality deals with coherent virtual object insertion into real environments with consistent lighting. As pointed by Jacobs and Loscos [**?**], major progresses were made, benefiting of improvements in acquisition systems or rendering methods. Early methods used manual light modeling (Fournier *et al.* [**?**], Drettakis *et al.* [**?**], Loscos *et al.* [**?**], Haller *et al.* [**?**]). Although, manually modeling light sources is a tedious and time consuming work, which is not suited for mobile usage.

Environment maps are a convenient way to capture and represent incident lighting. Debevec [**?**] was the first to describe the use of environment map as a light source model. He acquires an environment omnidirectional image using HDR imaging and a gaze ball. Sato *et al.* [**?**] used a similar rendering technique but acquire the environment with a couple of fisheye lens cameras. Numerous similar works were proposed (Agusanto *et al.* [**?**], Supan *et al.* [**?**], Grosch [**?**]) but all of them need complex calculations and/or specialized hardware, which is not suited for mobile implementation.

Potential solutions exists for handling complex lighting within an

*e-mail:hadrien.croubois@ens-lyon.fr
†e-mail:jean-philippe.farrugia@univ-lyon1.fr
‡e-mail:jean-claude.iehl@univ-lyon1.fr

mobile AR context. Snyder [**?**] propose explicit solutions for shading an object with extended light sources, providing that the BRDF follows a specific model (Lambert or Phong power-law model). This work is interesting but only applies on uniform light sources. Lighting with an environment map has been tackled by McGuire *et al.*[**?**] by exploiting the fact that modern graphic processors allows the filtering of cube-maps. Similarly to Snyder's work, this work assumes that the BRDF is a specific one (normalized Blinn-Phong). The main issue of this method is that lighting contributions are only integrated on one face of the cube-map, leading to visual artefacts on glossy surfaces when the cube-map faces are very different. Finally, Calian *et al.*[**?**] proposed an alternative by capturing shading instead of lighting with a specific probe with a custom geometry designed to fit different lighting computations. Although computation cost is low, the material of the virtual object has to be identical to the probe's, therefore limiting possible appearances. Furthermore, a specific target has to be built for every lighting configuration.

None of these techniques suit our needs for mobile AR: they are too restrictive or too computationally expensive, and all of them use intrusive probes or gaze balls. In this paper, we propose a method which dynamically capture an environment map with cellphone cameras and lights a virtual object with it in real time. We extended mcGuire's to take into account every face of the cube-map in the incident luminance integration. We demonstrate that extensions to anisotropic models are also possible. This method has been implemented on a high-end tablet using OpenGL ES 3.0 and runs at real-time framerates.

3 Contribution

This section will introduce our method for real-time mobile AR. We track the device with specifically designed frame markers using QRCodes. The surrounding lighting environment is dynamically captured and updated using the device's front camera. The virtual object is then directly lit with this environment map using an extended version of mcGuire*et al.*'s work[**?**]. Finally, projected shadows are approximated with sphere proxies. We will now go into details of these points.

3.1 Environment capture

Incident lighting is modeled using environment maps. Of course, one could always create these maps using offline methods: a large number of applications facilitate the creation of panoramas on mobile devices. For example, Cyclorama[**?**] uses the device's vibrator to automatically rotate on a slick surface while aligning successive captures. One can easily add a wide angle lens to obtain environment maps.

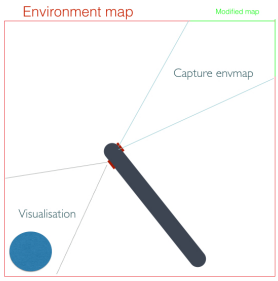

Figure 2: *Dynamic lighting capture.*

Although, offline capture assumes that the lighting environment is static, which is rarely the case in interactive applications. We propose to use the front camera to update the environment map. Since the camera is calibrated, and assuming the lighting is far enough, we may infer incident lighting direction for each pixel of a frame captured by the front camera and reproject it on the environment map, as shown on figure 2.

3.2 Image-Based Lighting

The ultimate goal of every rendering algorithm is the resolution of the rendering equation[**?**]. For any point p of our virtual object, the objective is to compute the output luminance from p in direction $\omega 0$:

$$L(p,\omega_0) = L_e(p,\omega_0) + \int_{\Omega+} f(p,\omega_0,\omega_i) L_i(p,\omega_i) cos\theta_i d\omega_i \quad (1)$$

with $L_e(p,\omega_0)$ being the emitted luminance, $f(p,\omega_0,\omega_i)$ being the reflectance function, $L_i(p,\omega_i)$ being the incident luminance and ω_i being the incident direction.

In our application, incident luminance L_i is represented with an environment map. For implementation purposes, this environment map is mapped on a cube (cube-map) around the virtual object.

The literature on how to compute environment lighting is abundant (see [**?**] for an overview). The regular way to evaluate this quantity is to sample the hemispherical domain to obtain point light sources across the environment map. However, this is too costly for mobile usage.

A faster solution is to approximate the convolution of the incident luminance by filtering the environment map. If the reflectance function is simple enough, or easily separable in simple components, the hardware filtering capacities of the graphics hardware may be used. MacGuire *et al.*[**?**] demonstrated that cosine power law filtering may be approximated by the mipmap levels of the cube-map. For a cosine power law with exponent s, the corresponding mipmap level m is given by (see paper for details):

$$m = \log(w\sqrt{3}) - 0.5 * \log(s+1) \quad (2)$$

Therefore, any reflectance function that has cosine power law components may be used with this method. MacGuire *et al.* propose to use the normalized Blinn-Phong reflectance model:

$$f(\omega_o,\omega_i) = \frac{1}{\pi}(k_L + k_G \frac{s+8}{8} max(\frac{\omega_i+\omega_o}{\|\omega_i+\omega_o\|}.n, 0)^s) \quad (3)$$

When replacing and integrating this reflectance function in the rendering equation, the diffuse $L_d(p,\omega_0)$ and glossy$L_g(p,\omega_0)$ components may be separated :

$$L(p,\omega_0) = L_e(p,\omega_0) + \frac{1}{\pi}(k_L L_d(p,\omega_0) + k_G L_g(p,\omega_0)) \quad (4)$$

with

$$L_d(p,\omega_0) = \int_{\Omega+} L_i(p,\omega_i) cos\theta_i d\omega_i \quad (5)$$

MacGuire *et al.* assume that the diffuse component Ld is proportional to the lowest level of the mipmap, and the glossy component is proportional to the sample in the mipmap level computed by

Figure 3: *Result of our image based lighting method. Left: glossy BRDF. Right: diffuse BRDF.*

equation 2. Therefore, computing luminance at point$p$ is just the combination of two texture fetches.

However, this method may lead to visible artefacts: for the diffuse component, a single face of the cube-map is chosen and is the only one to contribute to shading. If two adjacent faces of the cube-map are very different at the lowest level, a discontinuity will appear in the diffuse component of the object. We propose an improvement of this method to take into account every visible faces from point p.

To evaluate this, let us calculate the solid angle described by a partially visible face F:

$$\Omega_F(\vec{n}) = \iint_F \frac{Hs(\vec{\omega}.\vec{n})}{\|\vec{\omega}\|^3}\,\mathrm{d}\omega \tag{6}$$

with Hs being the Heavyside function. The contribution of each face may then be calculated by integrating the normalized dot product $\frac{\vec{\omega}.\vec{n}}{\|\vec{\omega}\|}$ with the solid angle on the whole face :

$$\begin{aligned} W_F(\vec{n}) &= \iint_F \frac{\vec{\omega}.\vec{n}}{\|\vec{\omega}\|} \times \frac{Hs(\vec{\omega}.\vec{n})}{\|\vec{\omega}\|^3}\,\mathrm{d}\omega \\ &= \iint_F \frac{\vec{\omega}.\vec{n} \times Hs(\vec{\omega}.\vec{n})}{\|\vec{\omega}\|^4}\,\mathrm{d}\omega \end{aligned} \tag{7}$$

Therefore, assuming incident diffuse light from face $\vec{F}$ $\mathcal{L}_d(\vec{F})$ and pre-computed ambiant occlusion factor $\mathcal{P}_\mathcal{V}(p)$ are known, the diffuse luminance $\mathcal{L}_d(p,\vec{n})$ of point P with a normal $\vec{n}$ is given by :

$$\mathcal{L}_d(p,\vec{n}) = \frac{\mathcal{P}_\mathcal{V}(p)}{\pi} \sum_{F \in Faces} \mathcal{L}_{id}(\vec{F}) W_F(\vec{n}) \tag{8}$$

Equation 8 is computed on each vertex and has to be rapidly evaluated. Since $W_F(\vec{n})$ depends roughly only on θ (θ and ϕ being euler angles), a numerical approximation may be obtained by a function parametrized by $\vec{n}.\vec{F} = \cos(\theta)$.

$$approx : \cos(\theta) \mapsto \frac{\left[\max\left(.75 + \cos(\theta), 0\right)\right]^2}{1.75} \tag{9}$$

Despite its simplicity, equation 9 is very close to $W_F(\vec{n})$ and may be used in equation 8 with reasonable precision.

3.3 Shadowing

There is numerous evidences that shadows enhance scene comprehension, especially on spatial relationships between objects (Madsen *et al.*[**?**], Sugano *et al.*[**?**]). Although, shadowing a complex geometry with complex lighting is not an easy task. For this purpose, we propose to replace the geometry with sphere proxies to simplify occlusion computation. We will assume that the local geometry around the virtual object may be approximated by a horizontal plane. The virtual object lies onto this plane, and shadows will be projected on it. To compute soft shadows, one has to evaluate environment visibility for every point of the horizontal plane. By replacing the actual geometry with sphere proxys, the obstructed part of the environment map may be calculated and incident lighting may be dimmed accordingly (see figure 4). Additional details are given in the supplemental material. Figure 3 shows some results of our image-based rendering method with shadows.

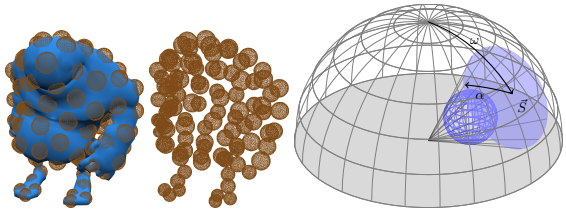

Figure 4: *Sphere decomposition of 3D objects and obstruction computation*

4 Results

4.1 Implementation details

This work has been developed in C++ using OpenCV 2.4 for tracking, OpenGL 3.0 for graphics and GLSL 3.0 for shaders. For conveniency reasons, it was initially developed for desktop PC under Linux and was later ported on iOS devices using OpenGL ES 3.0. The port was greatly facilitated by Objective C's compatibility with C++ and the similarity between OpenGL ES 3.0 and OpenGL 3.0 . The chosen target hardware is high-end iOS devices, like iPhone 5S and iPad Air, which are to this date the only Apple devices that can handle OpenGL ES 3.0. Although, porting the application on older devices should be possible by taking into account the differences between OpenGL ES 2.0 and 3.0.

4.2 Results with dynamic environment map capture

We will now demonstrate the capacity of our system to cope with dynamic lighting environments. Figure **??** shows a time lapse sequence of this ability. On the first frame, the virtual object is fully lit by the environment. On the second frame, the operator places his finger on the front camera of the device. One may see that the lighting modification is reported on the shading of the object. This application also runs on a iPad Air at approximately 25 frames per second, the frame resolution is 640×480. Additional results may be found in the supplemental material.

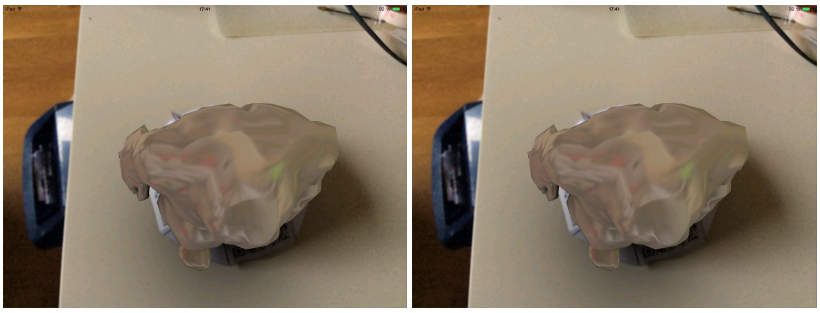

Figure 5: *Results with dynamically updating envmaps.*

5 Discussion and Possible Extensions

As said in section 3.2, the BRDF of the virtual object has to follow the Blinn-Phong model. This is the price to pay for real-time shading with complex light sources. A extension to micro-facet model BRDF seems doable by evaluating the anisotropic integration of the environment map, using hardware texture gradients for example. Although, at present time, texture gradients only apply on a single face of the cube-map and does not handle face continuity.

Another limitation of our method is that object lighting is represented by an environment map, naturally assuming distant lighting. An extension of this is possible like shown by Brennan[**?**] by using a crude approximation of the scene geometry (a simple bounding box or sphere may be sufficient) and adjusting the object's texture coordinates relatively to its position. With this method, the lighting is not considered infinitely distant anymore and the parallax effect is taken into account. Although, the lighting computation are more complex since the light direction is not the same for every point of the object, making our computations for shading in section 3.2 invalid.

Mobile implementation also presents some limitations due to hardware restrictions. First, shadow evaluation with sphere proxies is a little too costly for mobile devices: the decomposition of the geometry has to be limited to a reasonable number (around 50) of spheres to maintain an interactive framerate. Second, at present time, it is impossible to stream video from both cameras (front and back) at the same time on iOS, probably for efficiency and bandwidth reasons. To cope with this, we chose to mainly use the back-facing camera and to rapidly switch every five seconds on the front-facing camera to take a picture. Therefore, the environment map is only updated every five seconds and the video stream briefly freezes during this capture. Future updates of software and/or hardware will probably correct this issue.

Finally, in current implementation, captured images are low dynamic range (LDR). Strong light sources may not be identified in the environment map and no hard shadows will be rendered. High dynamic range (HDR) acquisition is a tough problem in real-time applications: the traditional method involves successive captures, possibly with quite long exposure times and naturally limiting the update rate of the environment map. Some works have been done on HDR acquisition with a single frame (Li *et al.*[**?**], Meylan *et al.*[**?**], Rempel *et al.*[**?**] but most of them are too slow or too approximative to fit mobile AR requirements. One potential solution for this problem may be to incorporate knowledge of the environment in the HDR reconstruction: by segmenting the LDR image in high luminance zones, one may identify potential light sources by knowing the context. For example, for interior scenes, chances are high that light sources are on the ceiling (light bulbs and neon tubes) and/or on the sides (windows). An artificial energy enhancement may be applied to these zones when reconstructing the HDR information.

6 Conclusion

A new solution for common illumination between real and virtual objects was presented in this paper. Assuming distant lighting, Blinn-Phong BRDF and planar local geometry, it handles both dynamic environments and soft shadows while being simple enough to run on a modern mobile device. It is achieved at real-time frame rates, leaving some computational ressources for other purposes. Finally, as shown in previous section, numerous extensions are possible.